**First Assignment**

**Name： 余韦藩 ID： 2020285102**

1. **Using the formula, briefly describe the process of a public key cryptography for digital signature and verification.**

**Answer:** Assume that Alice want to send the message m to Bob. The private key and public key of Alice is PrivateKey and PublicKey respectively.

1. First, Alice encryption the message m by SHA256 to obtain h that is **h=SHA256(m)**
2. Second, Alice use her private key to encryption the h and obtain r that is **r=AES(h, PrivateKey).** This process called “digital signature”.
3. Then Alice transmits **m** and **r** to Bob.
4. Bob search the public key of Alice online and decryption the message and obtain m’ that is **h’=AES(r, PublicKey)**.
5. Finally, Bob encryption the message m by SHA256 to obtain h that is **h=SHA256(m).** If the **h** and **h’** is same, it indicates that the message is sent by Alice. In other words, Bob verify successfully. Else, Bob verify fail.
6. **Using the formula, briefly describe the process of a public key cryptography for secret key transmission, and encryption and decryption of message m in conjunction with AES.**

**Answer:** Assume that Alice want to send the message m to Bob. The private key and public key of Alice is PrivateKey\_A and PublicKey\_A respectively. The private key and public key of Bob is PrivateKey\_B and PublicKey\_B respectively.

1. First, Alice use her private key to encryption the message m and obtain m’ that is **m’=AES(m, PrivateKey\_A)**.
2. Second, Alice searches the public key of Bob (**PublicKey\_B**) online use it to encryption her private key k’ that is **k’=Encoder(PrivateKey\_A, PublicKey\_B).** The Encoder is asymmetric encryption algorithm.
3. Then Alice transmits **m’** and **k’** to Bob.
4. Third, Bob decodes the private key of Alice that is k by **k=Decoder(k’, PrivateKey\_B).** The Decoder is asymmetric decryption algorithm.
5. Finally, Bob searches the public key of Alice online and decryption the message and obtain m that is **m=AES(m’, PublicKey\_A)**.
6. **Given 100 nodes, how to negotiate a ordering efficiently and fairly?**

**Answer:** In order to convenience to explain the negotiation process, we record 100 nodes as usr1,usr2,usr3,......,usr100.

1. At fist, all the users generate their own 256 bits random binary numbers which are recorded by xn1,xn2,xn3,......,xn100 respectively.
2. Once a user finishes generating his own 256 bits random number, then he provides the hash value corresponding to his 256-bit random number to inform others that he is ready. We record all the hash value of all the users as h1,h2,h3,......,h100.
3. Once all the 100 hash value are provided, all the users provide their 256 bits random number.
4. Next, we arrange the 100 random numbers obeying the sequence from small to large and links each other to form the sequence binary number (x1|x2|x3|......|x100).
5. Then calculate r=Hash(x1|x2|x3|......|x100).
6. After that we calculate the absolute distance of each 256-bit random number between r for each user. For example for the smallest random number usr1, the distance is |r－x1|. Then the node corresponding to the least distance has priority to pack a block.
7. **Given 100 nodes, how to negotiate a dynamic ordering efficiently and fairly. For example, after a node packs a block, then we know which node will pack the next block.**

**Answer:** In order to convenience to explain the negotiation process, we record 100 nodes as usr1,usr2,usr3,......,usr100.

1. At fist, all the users generate their own 256 bits random binary numbers which are recorded by x1,x2,x3,......,x100 respectively. Once a user finishes generating his own 256 bits random number, then he calculate the first hash value corresponding to his 256-bit random number and provide the second hash value to corresponding to first hash value to inform others that he is ready. We record all the first hash value of all the users as hn1,hn2,hn3,......,hn100 and record second hash value of all the users as hn1’,hn2’,hn3’,......,hn100’.
2. Once all the 100 second hash value are provided, all the users provide their first hash value.
3. Next, we arrange the 100 first hash number obeying the sequence from small to large and links each other to form the sequence binary number (h1|h2|h3|......|h100).
4. Then we calculate r=Hash(h1|h2|h3|......|h100) and calculate the absolute distance of each first hash value between r for each user. For example for the smallest first hash value usr1, the distance is |r－h1|. Then the node corresponding to the least distance has priority to pack a block. Next, we select the next user to pack a block. We assume that user1 get the priority to pack a block in the first round and user1 should public its 256-bit random number x1. Now, we calculate the r’=Hash(x1, r) and calculate the absolute distance of each first hash value between r’ for the rest of the user. Then the node corresponding to the least distance has priority to pack the next block.
5. In conclusion, we continuously re-calculate the r=Hash(xn, rn) (xn represents the last priority user and rn represents the last r) and calculate the absolute distance of each first hash value between r for the rest of the user. The node corresponding to the least distance has priority to pack the next block.